Um Recurso Educacional para Desenvolver a Habilidade da Percepção de Padrões de Equívocos com Aprendizes de Programação

Gabriel Sílverson Gomes
gabriel.silverson@ufms.br
Universidade Federal de Mato Grosso do Sul, Brasil

Ygor Takashi Nishi
takashi.nishi@ufms.br
Universidade Federal de Mato Grosso do Sul, Brasil

Paula Bosse Ramos
paula.bosse.ramos@gmail.com
Instituto Federal de Mato Grosso do Sul, Brasil

Leonardo Souza Silva
leonardo.silva@ufms.br
Universidade Federal de Mato Grosso do Sul, Brasil

Eduardo Guerra
guerraem@gmail.com
Free University of Bolzen-Bolzano, Itália

Igor Scaliante Wiese
igor@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Brasil

Yorah Bosse
yorah.bosse@ufms.br
Universidade Federal de Mato Grosso do Sul, Brasil

RESUMO

A aprendizagem de programação é vista como uma tarefa complexa, onde existem alunos com dificuldades em entender os conceitos, e outros que entendem mas possuem dificuldades em aplicá-los. As dificuldades juntamente com o desconhecimento da sintaxe das linguagens de programação, fazem com que aprendizes cometam erros durante o desenvolvimento de seus códigos. Pesquisadores analisaram e catalogaram os erros de programação mais frequentemente cometidos pelos aprendizes, os padrões de equívocos. A presença de padrões de equívocos podem tornar o aprendizado mais demorado, fazendo com que os alunos desistam da atividade que estão resolvendo ou até mesmo da disciplina. Nesse contexto, o objetivo geral deste trabalho é a construção de recurso educacional para desenvolver a habilidade da percepção de padrões de equívocos com aprendizes de programação, onde foi desenvolvido um conjunto de questões objetivas que abordam padrões de equívocos cometidos em C e em Python. Trabalhar com os alunos não só o que é correto, mas também os equívocos, poderá desenvolver essa habilidade tão importante para programadores, fazendo-os perceber os erros nos seus códigos, assim como corrigí-los. Neste estudo foram desenvolvidas 63 questões para trabalhar padrões de equívocos cometidos em C e 46 questões para Python. Essas questões estão disponíveis *online*, em diversos formatos, para professores e aprendizes utilizarem durante o ensino-aprendizagem de programação.

EduComp'23, Abril 24-29, 2023, Recife, Pernambuco, Brasil (On-line)

CCS CONCEPTS

• **Social and professional topics → CS1**.

PALAVRAS-CHAVE

Programação Introdutória, CS1, Aprendizado de Programação, Padrões de Equívocos, C, Python

1 INTRODUÇÃO

Preparar novas gerações de desenvolvedores profissionais e casuais é um grande desafio, conforme pode ser observado pelos índices de reprovações nas disciplinas de introdução à programação, chegando a aproximadamente 30% [3, 5, 15, 21, 37].

A programação e a informática, em geral, são transversais a muitas ciências exatas, naturais, sociais e humanas e o seu conhecimento poderá ser aplicado com benefícios em muitos dos seus domínios de atividade profissional e científica [27]. Nesse contexto, é cada vez mais necessário estender os conhecimentos de programação para além dos profissionais de informática [14].

O uso de linguagens de programação vem crescendo entre os profissionais e são muitos cursos de graduação, de diversas áreas, que estão incluíndo a disciplina de introdução à programação em suas grades curriculares [4]. Porém a programação de computadores é uma disciplina tradicionalmente difícil para estudantes iniciantes [29].

Segundo diversas pesquisas é possível afirmar que um dos problemas encontrados é a quantidade de novos conceitos e práticas necessárias à proficiência em programação, dificultando o aprendizado dos recém-chegados à área [10, 29, 33]. A maioria dos alunos de disciplinas de programação, sentem dificuldades de aprender a programar, pois possuem poucos paralelos com suas experiências prévias na educação básica. Por esse motivo, observa-se um alto nível de evasão e abandono nesses cursos [33].

Além disso, alguns alunos reportam não entender os conceitos abordados na disciplina de programação [18, 24, 30], enquanto outros dizem entender os conceitos, mas não sabem como utilizá-los na construção dos algoritmos [16, 25]. São vários tipos de erros cometidos pelos aprendizes [6], como os que interrompem a execução do programa por se tornar incompreensível para o compilador e outros que permitem a execução porém levam a um resultado indesejado pelo uso incorreto de comandos. Hristova et al. [19] identificaram alguns erros como a troca do = por ==; & por && e do | por ||; assim como os separadores errados na estrutura de repetição 'for'. Porém são poucos estudos que fazem algum tipo de classificação dessas dificuldades, entre os que fazem podemos citar [6, 9, 19, 34].

Tornar a aprendizagem de programação mais branda é muito importante. Segundo Robins et al. [31], os estágios iniciais da aprendizagem são críticos para o resultado do processo (teoria do Momento de Aprendizagem – *Learning Edge Momentum* – LEM), já que, uma vez estabelecido um momento negativo, é difícil superá-lo. Para os autores, adquirir um conceito com sucesso torna o aprendizado de outro conceito estreitamente vinculado mais fácil.

Apesar de que as dificuldades sempre existirão, é importante gerenciar a sua frequência e intensidade, fazendo com que os aprendizes aprendam com seus erros [13, 17, 35]. O presente estudo tem como objetivo geral a construção de recurso educacional para desenvolver a habilidade da percepção de padrões de equívocos com aprendizes de programação.

O recurso educacional proposto é composto de questões de múltipla escolha que abordam equívocos frequentemente cometidos pelos aprendizes em seus códigos nas linguagens C e Python. Essas questões abordam erros cometidos em diversos tópicos estudados na disciplina de introdução à programação, o que auxiliará o professor a trabalhar as dificuldades com seus alunos. Levando-se em consideração os 41 antipadrões catalogados em Bosse et al. [6], foram criadas 63 questões abordando os equívocos cometidos com a linguagem C e 46 com Python, totalizando 109. Criamos um *site* onde foram disponibilizadas todas questões. As questões estão disponíveis de forma individual, em diversos formatos, entre eles em PDF, DOCX e o formato para *upload* no Moodle. Por meio de questões de múltipla escolha, apresentando os erros mais frequentemente cometidos pelos aprendizes, acredita-se que os alunos irão desenvolver a habilidade de percepção de padrões de equívocos, evitando ou então identificando e corrigindo os equívocos mais rapidamente.

2 TRABALHOS RELACIONADOS

As matérias de introdução a programação são apresentadas aos alunos logo nos primeiros semestres na maioria das grades curriculares de cursos relacionados a computação [4], fase em que os alunos estão passando por um momento de adaptação à vida na universidade [20]. A maioria desses alunos possuem pouco ou nenhum contato com os conceitos como lógica e criação de algoritmos, o que pode dificultar o aprendizado de programação [6, 9, 19, 34], acarretando em altos níveis de evasão nesses cursos [33].

O aprendizado de programação é uma tarefa complexa [20, 21], sendo um processo lento e gradual de transformar o "novo em familiar" [11], ninguém aprende programação sem investir tempo de treinamento intensivo [16].

2.1 Antipadrões

Para Brown et al. [7], antipadrão é uma forma literária que descreve uma solução comum a um problema que gera consequências decididamente negativas. Os antipadrões podem ser considerados os resultados da falta de conhecimento ou aplicação de um padrão bom em um contexto errado. Os antipadrões da programação tem o mesmo conceito que o termo "padrões de equívocos na aprendizagem de programação", ou na forma reduzida "padrões de equívocos", que será usado neste estudo. O termo padrão segundo Alexander et al. [1], "descreve um problema que ocorre repetidamente em nosso ambiente e, em seguida descreve também o núcleo da solução para esse problema, de forma que você possa usar essa solução um milhão de vezes, sem que o problema ocorra novamente".

Conhecer erros frequentemente cometidos em códigos, os antipadrões, também é importante para auxiliar alunos, professores e pesquisadores, e estudá-los é uma importante atividade de pesquisa [7]. Não basta conhecer os padrões "bons" existentes nos sistemas bem sucedidos, e sua inexistência em sistemas mal sucedidos. É importante mostrar a presença de antipadrões em sistemas mal sucedidos e sua ausência em sistemas bem sucedidos [7].

Achar erros, especialmente no seu próprio código é uma dificuldade que os alunos enfrentam [12, 36]. O uso das metodologias de estudo de programação não são efetivas, como o caso da estratégia de memorização, que é a leitura e visualização de exercícios resolvidos [16]. Em contrapartida, Lahtinen et al. [23] afirmam que parte dos estudos deve ser a prática, para aprender fazendo. Segundo Mhashi and Alakeel [26], uma abordagem usada para aprimorar o conhecimento dos alunos e desenvolver suas habilidades é aumentar o número de exercícios, selecionando-os cuidadosamente e resolvendo-os posteriormente. Resolvendo os exercícios posteriormente com os alunos gera uma oportunidade de esclarecer dúvidas. Para Gomes and Mendes [16], trabalhar em cima das dificuldades que são frequentemente encontradas poderá auxiliar e tornar o aprendizado mais fácil.

2.2 Tipos de padrões de equívocos

Segundo Hristova et al. [19] os erros na aprendizagem de programação podem ser classificados em três tipos: (i) os de sintaxe, (ii) os de semântica e (iii) os de lógica. Os erros de sintaxe são os detectados pelo compilador pela escrita errada dos comandos. Os pesquisadores afirmam que as mensagens de erro mostradas pelo compilador não apontam necessariamente a direção correta para a solucão do problema.

Os erros de semântica ocorrem quando o código passa pelo compilador ou interpretador sem erros, mas o programa não executa o que o desenvolvedor esperava, gerando um resultado errado. E os erros mais genéricos foram chamados pelos pesquisadores de erros de lógica [19]. Os pesquisadores que catalogaram antipadrões em C e Python [6] adotaram uma abordagem diferente para classificar os erros que não são de sintaxe e nem de semântica, ao qual denominaram de erros de estilo. Um dos erros catalogados nesse tipo foi a falta de indentação em códigos escritos em C. Os erros de estilo não afetam a execução do programa, mas podem dificultar a leitura e a manutenção do código. Independente do nome dado aos tipos de erros, eles podem dificultar muito o ensino-aprendizado de programação.

3 MÉTODOS

O objetivo geral do trabalho é a construção de recurso educacional para desenvolver a habilidade da percepção de padrões de equívocos com aprendizes de programação. A Figura 1 mostra o passo a passo do trabalho realizado, descrito abaixo.

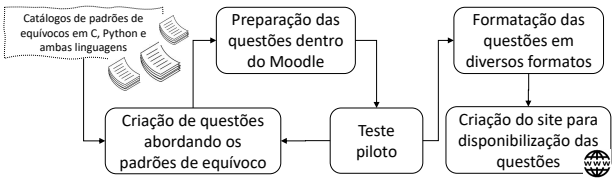


Figura 1: Metodologia.

Para conseguirmos utilizar os padrões de equívoco de uma forma que possam auxiliar na aprendizagem do aluno, foi elaborado um recurso de ensino-aprendizagem para aplicar esses padrões em questões de múltipla escolha. Cada questão proposta descreve um problema (enunciado), em conjunto com seguimento de código que contém um padrão de equívoco. Para alcançarmos o objetivo de aprendizado, desenvolvendo a habilidade de percepção dos padrões de equívocos, cada alternativa possui um feedback, explicando o motivo da resposta escolhida pelo aluno estar errada ou não, e um feedback geral que explica melhor o padrão de equívoco presente na questão.

A sequência dos conteúdos abordados nas questões segue a dos antipadrões. Os trechos de códigos que são apresentados para serem analisados foram desenvolvidos por alunos de turmas reais, com pequenas adaptações para que fiquem mais fáceis para os aprendizes interpretarem.

Segundo Buchweitz [8] o uso de questões de múltipla escolha pode facilitar a abordagem do conteúdo, pois pode-se ter um escopo abrangente dos conceitos administrados. Optamos utilizar essa estratégia pois conseguimos tratar os três tipos de erros (Sintático, Semântico e de Estilo) nas linguagens escolhidas de forma objetiva, fazendo o aluno raciocinar sobre cada um dos padrões de equívocos catalogados.

As questões foram trabalhadas para os erros cometidos em C e Python. Essa escolha das linguagens se deu por vários motivos. O motivo principal é que usamos os 41 padrões de equívocos catalogados em Bosse et al. [6] para desenvolver as questões e esse catálogo é de erros cometidos nessas duas linguagens. Os 41 padrões de equívocos e o ID das questões geradas pra cada um deles, com seus respectivos níveis de dificuldades, podem ser visto na Tabela 1. Além disso, as linguagens C e Python são classificadas entre as mais populares, utilizadas em diversas universidades [2, 32]. E para finalizar, são linguagens adotadas nas disciplinas de introdução a programação da Universidade Federal de Mato Grosso do Sul (UFMS), campus de Ponta Porã (CPPP), onde o recurso educacional foi aplicado numa pequena amostra.

O nível de dificuldade das questões acompanha o nível de complexidade dos tópicos abordados nos cursos de introdução à programação, como mostra a Tabela 2. As questões que abordam problemas com variáveis, por exemplo, terão um nível de dificuldade menor do que questões com problemas de funções ou matrizes. Além disso, uma mesma questão pode abordar mais de um conteúdo e equívoco, o que aumenta sua dificuldade também. A meta é que o professor tenha questões para serem utilizadas do início ao final do curso. Para o teste piloto, as questões foram preparadas no Moodle e semanalmente foram disponibilizadas cinco delas para os alunos resolverem, dando a nós a possibilidade de ajustes caso necessário.

Finalizando a criação das questões e o teste piloto, as questões foram organizadas em tabelas, gerando um banco de questões, que está disponibilizado no *site* (https://antipatterns.netlify.app/) em diversos formatos para serem baixados, como Aiken, GIFT, Moodle XML e XHTML, PDF, DOCX, além da opção de visualização no própria site. As tabelas possuem campos como: linguagem de programação utilizada, ID do padrão de equívoco, ID da questão, nome da questão, descrição da questão com a imagem auxiliar e alternativas criadas, campo indicando alternativa correta, tipo de erro presente na questão, *feedback* geral e de cada alternativa. Na Figura 2 podemos observar um modelo da tabela onde as questões estão armazenadas.

QUESTÃO EM (LINGUAGEM DE PROGRAMAÇÃO)						
PADRÃO DE EQUÍVOCO						
ID	**NOME DO PADRÃO DE EQUÍVOCO**					
***	NOME					
QUESTÃO						
ID da QUESTÃO	**TÍTULO DA QUESTÃO**					
***	TÍTULO					
Problema: .... (IMAGEM) a. . b. . c. . d. .						
Alternativa correta: (.)						
TIPO DE ERRO:	.	Sintaxe		Semântica		Estilo
Conteúdos necessários para entender o código: A: ..... B: ..... C:						
Feedback geral:						
Feedback sobre as respostas: a. **Parabéns você acertou!** b. ..... c. ..... d.						

Figura 2: Modelo da tabela das questões.

A seguir será descrito o recurso educacional com mais detalhes.

4 RECURSO EDUCACIONAL

Criamos 63 questões abordando os equívocos cometidos com a linguagem C e 46 com Python, conforme Tabela 1. Foram abordadas

ID	ID Pad.Equ.	Nome do Padrão de Equívoco	ID das Questões (nível de dificuldade)	Linguagem	Tipo do Erro
1	C_G2	Falta de " ; " no final da linha	1-ABC01 (5), 1-ABC02 (5)	C	Sintaxe
2	C_GL1	Falta chamada da biblioteca	2-ABC01 (5), 2-BHJ02 (18), 2-BCD03 (7)	C	Sintaxe
3	C_IF1	Falta do "&" na frente da variável no "scanf"	3-ABC01 (5), 3-BDH02 (12), 3-BCD03 (7)	C	Semântica
4	C_IF6	Falta vírgula para separar o primeiro do segundo parâmetro no "scanf"	4-BDH01 (12),4-BFG02 (13)	C	Sintaxe
5	C_OF1	Uso indevido do "&" na frente da variável no "print"	5-ABC01 (5), 5-BCF02 (9)	C	Semântica
6	C_OF3	Escrita incorreta do comando "printf"	6-DFG01 (14),6-CDH02 (12)	C	Sintaxe
7	C_OF5	Parâmetro que identifica o tipo de dado de saída incorreta ou inexistente	7-BCG01 (10),7-ACI02 (11)	C	Sintaxe
8	C_OF6	Uso de "&" no lugar do "%" na função de saída de dados	8-BC01 (4), 8-DFH02 (15), 8-AFG03 (12)	C	Semântica
9	C_OF7	Não apresenta o resultado do usuário	9-BDK01 (15), 9-CDG02 (11)	C	Semântica
10	C_AE1	Tipo do resultado float em uma divisão de inteiros	10-ABC01 (5), 10-ACD02 (6)	C	Semântica
11	A_AE2	Fórmula aritmética errada	11-DFG01 (14), 11-EGK02 (20)	C	Semântica
12	C_AE3	Realizar o cálculo antes de ter valores nas variáveis utilizadas	12-BCD01 (7), 12-DHJ02 (19)	C	Semântica
13	C_RE5	Realizar comparação antes de ter valores nas variáveis utilizadas	13-AFG01 (12), 13-GHI02 (21)	C	Semântica
14	C_SS1	Uso incorreto das chaves de abertura ou de fechamento do "if" ou do "else"	14-CFG01 (13), 14-DFG02 (14)	C	Sintaxe
15	C_SS4	Colocar indevidamente ";" após a condição "if" e/ou após o "else"	15-BDG01 (11), 15-FG02 (11)	C	Semântica
16	C_RS3	Imprimindo resultado em local errado	16-ACI01 (11), 16-GHI02 (21)	C	Semântica
17	C_F1	Falta do return	17-DFG01 (14), 17-FGK02 (21)	C	Semântica
18	C_F2	Falta de abertura "{" e/ou fechamento "}" da função	18-DK01 (13), 18-HIJ02 (24)	C	Sintaxe
19	C_F6	Chamada incorreta de uma função tipada	19-CDK01 (15), 19-BDK02 (15)	C	Semântica
20	C_F9	Declaração incorreta dos parâmetros da função	20-CIK01 (20), 20-DK02 (13)	C	Sintaxe
21	C_F10	"return" x "printf"	21-HIJ01 (24), 21-HIK02 (25)	C	Estilo
22	P_V2	Usar palavra reservada para nome de variável	22-ABC01 (5), 22-ABC02 (5), 22-ABC03 (5), 22-ABC04 (5)	Python	Sintaxe
23	P_V4	Conversão incorreta de tipo de dado	23-ABC01(5), 23-AGK02(17)	Python	Sintaxe
24	P_V5	Falta conversão de tipo	24-ABC01 (5), 24-ABD02(6)	Python	Semântica
25	P_IF4	Falta parênteses no "input"	25-ABC01 (5), 25-ABC02 (5)	Python	Semântica
26	P_OF5	"print" seguido por "=" ou outro parâmetro incorreto	26-ABC01 (5), 26-ADK02 (14)	Python	Sintaxe
27	P_AE2	Operador aritmético errado	27-BCD01 (7), 27-BCD02 (7)	Python	Sintaxe
28	P_SS1	Falta dos ":" no final da linha do "if" ou do "else"	28-ADG01 (10), 28-BCD02 (7)	Python	Sintaxe
29	P_RS1	Ordem incorreta dos comandos na estrutura	29-FHI01 (20), 29-CFH02 (14)	Python	Semântica
30	P_SRS1	Uso de estruturas de repetição onde deveria ser de seleção	30-FHI01 (20), 30-FGH02 (18)	Python	Semântica
31	P_MDA1	Criação errada das linhas da matriz	31-HIJ01 (24), 31-HIJ02 (24)	Python	Semântica
32	P_F8	Função criada, porém, não chamada	32-FJK01 (24), 32-GHK02 (23)	Python	Semântica
33	G1	Falta de indentação	33-CFG01 (13), 33-CFG02(13), 33-ABC04 (5), 33-CDJ06 (14)	Python	Py = Sintaxe
			33-ABC03 (5), 33-CDJ05 (14)	C	C = Estilo
34	V1	Uso de variável inexistente	34-DGH01 (16), 34-BCD06 (7), 34-DGH02 (16), 34ABC04 (5)	Python	Sintaxe
			34-ABC03 (5), 34-BCD05 (7)	C	
35	V3	Atribuição usando "==" ao invés de "="	35-CFI01 (15), 35-BCD02(7)	Python	Py = Sintaxe
			35-CFI03 (15), 35-BCD04(7)	C	C = Semântica
36	IF2	Falta das aspas na chamada da função de entrada de dados	36-ACG01 (9), 36-DH02 (10)	Python	Sintaxe
			36-ACG03 (9), 36-DH04 (10)	C	
37	OF2	Falta das aspas na função de saída de dados'	37-ABI01 (11), 37-CDK02 (15)	Python	Sintaxe
			37-ABI03 (11), 37-CDK04 (15)	C	
38	OF4	Falta da vírgula para separar o primeiro do segundo parâmetro na função de saída de dados	38-HIJ01 (24), 38-EFG02 (15)	Python	Sintaxe
			38-HIJ03 (24), 38-EFG04 (15)	C	
39	RE1	O uso do "=" ao invés de "=="	39-DFG01 (14), 39-DFG02 (14)	Python	P = sintaxe
			39-DFG03 (14), 39-DFG04 (14)	C	C = semântica
40	SS2	O não uso do "else" onde seria adequado	40-EGH01 (17), 40-EGI02 (18)	Python	Estilo
			40-EGH03 (17), 40-EGI04 (18)	C	
41	RS2	Variável de controle não está sendo alterada	41-EGH03 (17), 41FHI04 (20)	Python	Semântica
			41-EGH03 (17), 41FHI04 (20)	C	

Tabela 1: Tabela dos padrões de equívoco catalogados nas linguagens C e Python.

ID	Conteúdos Abordados	Nível de Dificuldade
A	Variáveis (declaração, atribuição, uso)	1
B	Funções de Entrada de Dados	2
C	Funções de Saída de Dados	2
D	Expressões Aritméticas (+, -, /, *, ...)	3
E	Expressões Lógicas (AND, OR, NOT)	4
F	Expressões Relacionais (>, >=, <, <=, ==, !=)	5
G	Estruturas de Seleção (if, if..else, switch)	6
H	Estruturas de Repetição (while, for, do..while, ...)	7
I	Vetores	8
J	Matrizes	9
K	Funções	10

Tabela 2: Conteúdos abordados.

nestas questões 41 padrões de equívocos catalogados de códigos desenvolvidos por aprendizes. Cada questão foi organizada dentro de uma tabela que será explicada abaixo, por partes, usando como exemplo nas imagens uma questão que trabalha o padrão de equívoco C_IF1, ou seja, um erro de função de entrada "*scanf*" cometido na linguagem C.

Na primeira parte da tabela temos a informação de qual linguagem de programação será abordada na questão, em seguida vem as informações do padrão de equívoco e da questão, ambos com ID e nome/título (Tabela 3).

QUESTÃO EM C	
PADRÃO DE EQUÍVOCO	
ID	NOME DO PADRÃO DE EQUÍVOCO
C_IF1	Falta do "&" na frente da variável no "scanf"
QUESTÃO	
ID da QUESTÃO	TÍTULO DA QUESTÃO
3-BCD03	Média das provas

Tabela 3: Primeira parte da tabela das questões: identificação da linguagem, do padrão de equívoco e da questão.

O ID da padrão de equívoco foi retirado do catálogo que usamos como base [6]. O ID da questão segue o seguinte padrão:
(Número do padrão de equívoco) – (No mínimo uma letra, ou no máximo 3, referentes aos conteúdos abordados na questão) (número da questão)

Onde o número do padrão de equívoco é o ID existente na Tabela 1 e as letras são referentes aos conteúdos abordados na questão, listados na Tabela 2. No exemplo da Tabela 3 temos a questão **3–BCD03**, ou seja, uma questão que está trabalhando com o padrão de equívoco 3 (Falta do "&" na frente da variável no "scanf"), e que aborda três conteúdos (BCD) que são respectivamente, funções de entrada dados, funções de saída de dados e expressões aritméticas (+, -, /, *, ...), sendo esta a terceira (03) questão desse padrão de equívoco no banco de questões. Além disso, na Tabela 1, cada ID da questão é seguido por um parênteses com um número que identifica o nível de dificuldade dela. No caso da questão **3–BCD03**, o nível de dificuldade é sete: 2 (B - Funções de Entrada de Dados) + 2 (C - Funções de Saída de Dados) + 3 (D - Expressões Aritméticas (+, -, /, *, ...).

Em seguida vem o enunciado do problema, juntamente com a imagem do código que deveria resolver o problema exposto (Tabela 4). Esse código contém o equívoco que o aluno deverá identificar para escolher corretamente uma das alternativas que responde a questão.

Problema: Um aluno estava implementando um código para fazer a média das notas de sua matéria preferida, porém algo deu errado e o resultado não está aparecendo, analise o código e aponte o erro:

```
#include <stdio.h>

int main(){

    float media = 0, n1, n2, n3;

    printf(" Digite a nota da sua primeira prova: ");
    scanf("%f", &n1);

    printf(" Digite a nota da sua segunda prova: ");
    scanf("%f", &n2);

    printf(" Digite a nota da sua terceira prova: ");
    scanf("%f", n3);

    media = (n1 + n2 + n3) / 3;

    printf("\n Sua media final foi: %f ", media);

    return 0;
}
```

Tabela 4: Segunda parte da tabela das questões: enunciado do problema e código.

Em todas as questões serão mostrados trechos de códigos para o aluno, para ele analisar. Para uma melhor leitura dos códigos as cores das imagens foram escolhidas com base nos estudos sobre contraste, segundo Kulpa et al. [22]: "A cor é considerada o elemento visual da interface que influencia diretamente na qualidade da apresentação das informações transmitidas". A ferramenta para escrever os trechos de códigos é um editor online chamado Carbon.now.sh que tem a finalidade de gerar imagens de código de diversas linguagens e temas.

Na terceira parte da tabela aparecem as alternativas que serão apresentadas para o aluno, seguidas da identificação da alternativa correta. Além disso, aparece a identificação do tipo de erro e dos conteúdos abordados na questão (Tabela 5).

E para finalizar, aparecem os *feedbacks*. Eles têm o objetivo de auxiliar o aluno a entender o porque ele errou ou o porque as outras alternativas estavam erradas, juntamente com um *feedback* geral da questão. A Tabela 6 contém um exemplo de como são esses

a. O aluno esqueceu de colocar o '&' na frente da variável 'n3' no 'scanf'; b. A variável 'media' está sendo declarada de forma errada; c. O 'return 0' não precisa do ' ; ' por representar o final do código. d. A identação está incorreta.						
Alternativa correta: (a)						
TIPO DE ERRO:	.	Sintaxe	X	Semântica		Estilo
Conteúdos necessários para entender o código: B: Funções de Entrada de Dados C: Funções de saída de Dados D: Expressões aritméticas (+, -, /, *, ...)						

Tabela 5: Terceira parte da tabela das questões: alternativas disponíveis; alternativa correta; tipo de erro e conteúdos abordados.

feedbacks. Em uma turma com a aplicação constante desse material a percepção do aluno tende a melhorar.

Feedback geral: Por ser considerado um erro de semântica, o compilador não mostra erro na execução do código, porém ele não completa a execução e para. Temos que tomar cuidado com esse tipo de erro, pois em um código muito extenso a identificação do problema pode levar tempo.
Feedback sobre as respostas: a. **Parabéns você acertou!** b. Alternativa incorreta, quando estamos trabalhando com variáveis que irão receber um valor de outra variável, ou uma soma/divisão ou multiplicação, é recomendado sempre iniciar a variável com 0, pois assim temos certeza do que está nela, como a alocação das variáveis são feitas na memória, pode acontecer da variável receber alguma informação (lixo de memória) e acabar atrapalhando o resultado. c. Alternativa incorreta, em C para representar o fim de uma instrução devemos utilizar o ' ; '. d. Alternativa incorreta, como podemos observar no código acima, as instruções estão definidas de forma correta.

Tabela 6: Quarta parte da tabela das questões: *Feedback's*.

Figura 3: Imagem do site onde as questões estão disponíveis: Tela inicial.

Todas as questões foram disponibilizadas no *site* (https:// antipatterns.netlify.app/) para serem livremente usadas por professores e aprendizes (Figuras 3 e 4). Cada questão poderá ser localizada pelo conteúdo que se deseja trabalhar. A visualização poderá ser feita diretamente no *site* ou ser baixado para realização de atividades. Os formatos disponíveis são Aiken, GIFT, Moodle XML e XHTML, PDF e DOCX.

4.1 Implementação na Plataforma Moodle

Pensando numa forma de utilizar as questões de forma prática, podendo ser usadas com diversas turmas, chegou-se a plataforma Moodle pois é utilizada por diversas universidades. Segundo Okada et al. [28], o Moodle tem a maior participação de mercado internacional, com 54% de todos os sistemas de apoio *online* ao ensino e aprendizado. A plataforma Moodle é uma ferramenta de aprendizagem baseada na *WEB*, que pode ser utilizada para criar cursos virtuais, com recursos de comunicação, avaliação e gerenciamento de conteúdo.

Começamos preparando as questões dentro da plataforma e já nos primeiros testes obtivemos boas impressões. Ela é de fácil utilização e para o nosso contexto, ela tem diversos recursos que poderíamos utilizar, permitindo, por exemplo, o uso de questões objetivas e a inserção dos *feedbacks* para cada alternativa. Além disso, a plataforma Moodle é gratuita e de código aberto, o que permite que qualquer pessoa possa contribuir com o desenvolvimento da ferramenta.

Um dos primeiros recursos a ser explorado foi a criação dos cursos e questionários, e a ferramenta já disponibilizava uma série de modelos que poderiam ser utilizados, o que facilitou bastante o processo de criação dos mesmos. Além disso, a plataforma também disponibiliza uma série de *plugins* que podem ser utilizados para adicionar novas funcionalidades, como por exemplo, o *plugin* de gráficos e o de *feedback* automático.

O *plugin* de gráficos auxilia na hora de gerar relatórios de desempenho de cada aluno, possibilitando ver a média de notas da turma, quais questões foram mais acertadas, o tempo gasto, entre diversos outros. Além disso, podemos gerar relatórios por aluno, como mostra a Figura 5, que mostra os dados de um aluno que participou do nosso teste piloto.

Segundo Mhashi and Alakeel [26], um dos fatores relacionados com as dificuldades enfrentadas pelos alunos é a falta de *feedback* sobre as resoluções dos exercícios. Quando se está utilizando o recurso de *feedbacks*, o aluno visualiza o resultado imediatamente, recebendo o conhecimento do porque ele errou, como podemos ver na Figura 6. E, errando ou acertando a resposta, será exibido o *feedback* geral da questão no intuito de que o aluno entenda o conceito e o padrão de equívoco da questão, conforme mostrado na Figura 7.

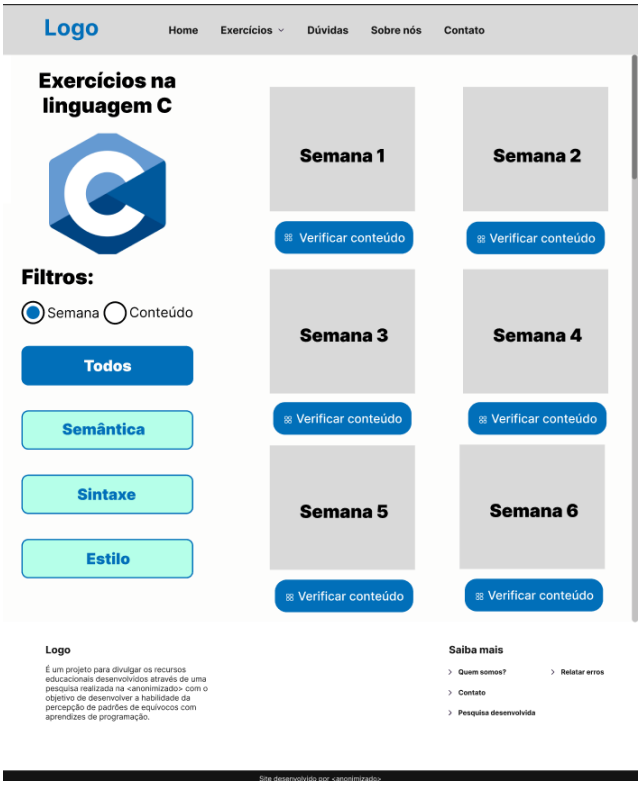
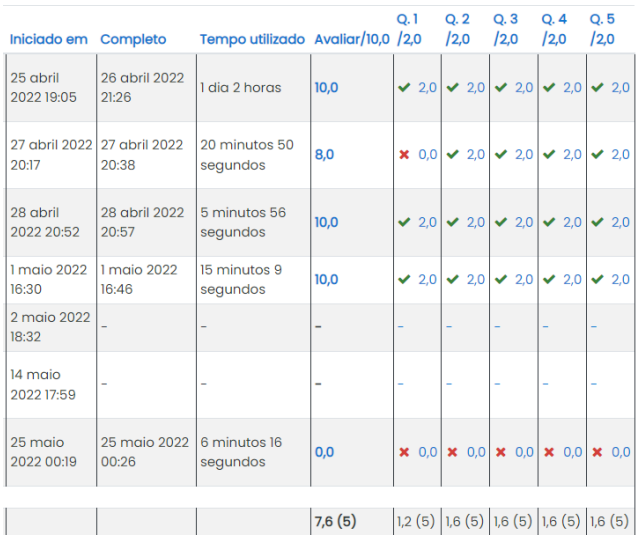


Figura 4: Imagem do site onde as questões estão disponíveis: Tela de seleção de conteúdo.



Iniciado em	Completo	Tempo utilizado	Avaliar/10,0	Q. 1 /2,0	Q. 2 /2,0	Q. 3 /2,0	Q. 4 /2,0	Q. 5 /2,0
25 abril 2022 19:05	26 abril 2022 21:26	1 dia 2 horas	10,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0
27 abril 2022 20:17	27 abril 2022 20:38	20 minutos 50 segundos	8,0	✖ 0,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0
28 abril 2022 20:52	28 abril 2022 20:57	5 minutos 56 segundos	10,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0
1 maio 2022 16:30	1 maio 2022 16:46	15 minutos 9 segundos	10,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0	✔ 2,0
2 maio 2022 18:32	-	-	-	-	-	-	-	-
14 maio 2022 17:59	-	-	-	-	-	-	-	-
25 maio 2022 00:19	25 maio 2022 00:26	6 minutos 16 segundos	0,0	✖ 0,0	✖ 0,0	✖ 0,0	✖ 0,0	✖ 0,0
			7,6 (5)	1,2 (5)	1,6 (5)	1,6 (5)	1,6 (5)	1,6 (5)

Figura 5: Relatório individual dos alunos por questões.

Com recurso de *feedbacks* automáticos, o aprendiz aprenderá a identificar o erro e como solucioná-lo.

E por último, vimos que a ferramenta possui um recurso de exportar os bancos de questões em formatos específicos da plataforma

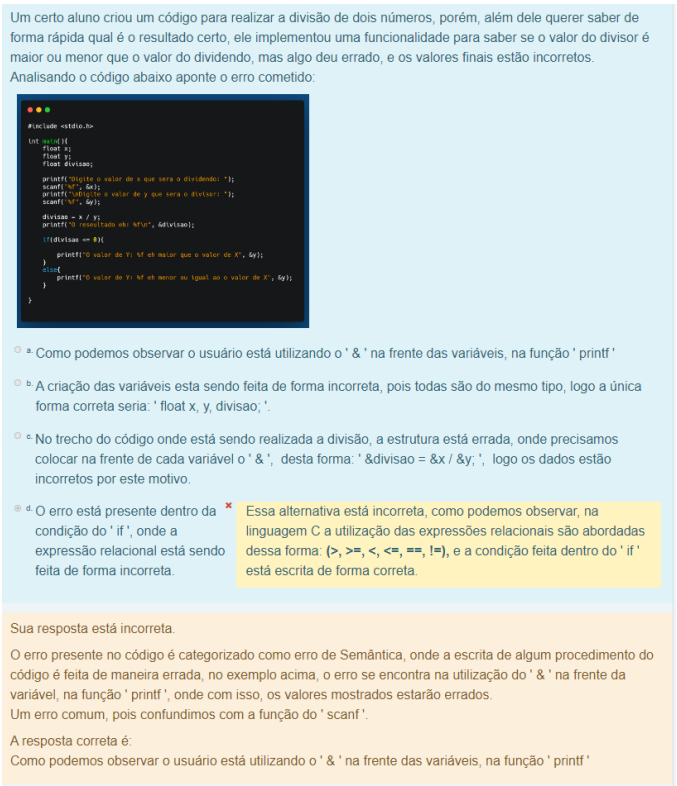


Figura 6: Feedback resposta incorreta.

Moodle e de outras plataformas de ensino, ou seja, o material desenvolvido pode ser compartilhado com outras instituições de ensino e consequentemente facilitará a disseminação do material. Essa ferramenta de exportação do banco de questões pode ser visualizada na Figura 8, onde são mostrados também os tipos de arquivos que estão disponíveis. Após a organização da plataforma, selecionamos um grupo de alunos para testar a ferramenta, e após a análise dos resultados, foi possível perceber que a ferramenta atendeu as expectativas.

5 TESTE PILOTO

Para testar as questões e a aplicação através da plataforma Moodle, convidamos alunos voluntários de uma turma de programação introdutória da UFMS-CPPP. O professor da turma estava ensinando com a linguagem de programação C, logo todas as questões e padrões de equívocos usados no teste piloto foram com essa linguagem. Foi criado um grupo no aplicativo de mensagens *WhatsApp* com os alunos que aceitaram participar. Nesse grupo eram divulgados semanalmente a disponibilização de questionários com 5 questões abordando os padrões de equívocos em conteúdos que eles estavam aprendendo em aula. Escolhemos aplicar apenas cinco questões objetivas semanalmente para evitar que o teste tomasse muito tempo do aluno, visto que esse processo se repetiria toda semana.

Após uma semana com o questionário em aberto para que os alunos respondessem, ele ficava indisponível e era colocado outro questionário abordando novos conteúdos. Em caso de alta taxa de

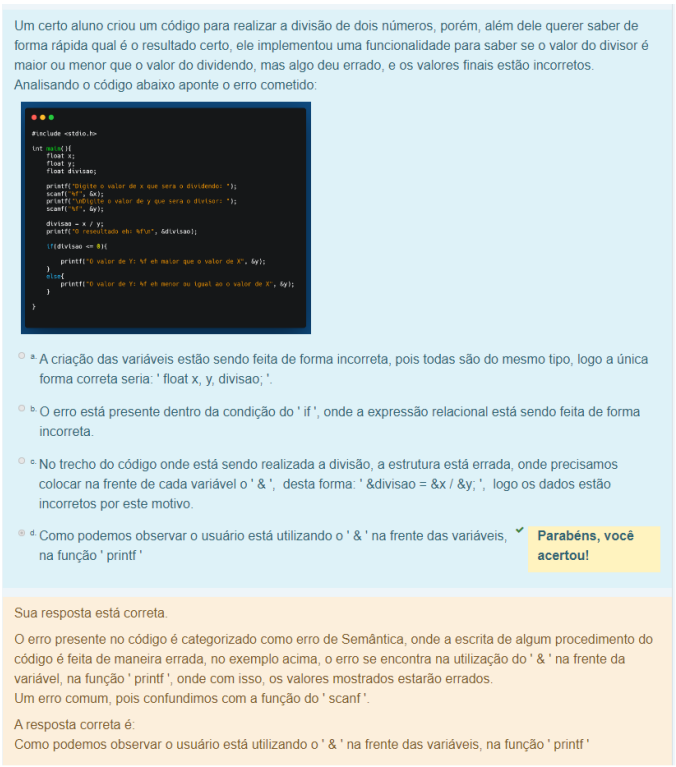


Figura 7: Feedback resposta correta.

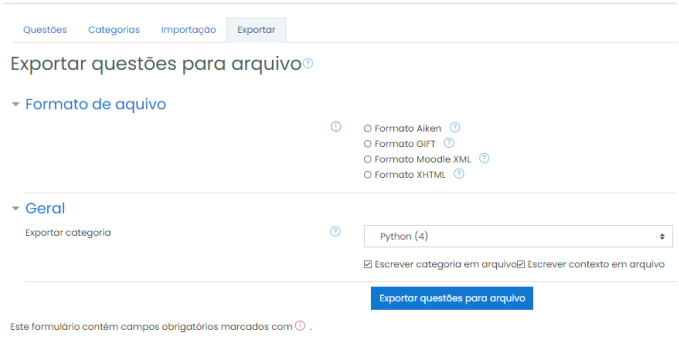


Figura 8: Ferramenta de exportação de questões.

erro em um padrão de equívoco, ele era incluído novamente em questionários posteriores.

Nesse teste piloto obtivemos um total de 22 respostas durante todo o período de análise. Dos alunos inscritos na disciplina, obtivemos uma taxa de 20% de participação durante todo o período. Todos os questionários foram disponibilizados no Moodle da UFMS, logo, além dos avisos no grupo do *WhatsApp*, o próprio sistema notificava os alunos.

Nas nossas análises observamos que muitas das vezes os alunos entravam no questionário, porém não respondiam todas as questões. Além disso, durante o avanço da pesquisa, o número de participantes foi diminuindo, em comparação a primeira semana, assim como o índice de respondentes de todas questões por questionário. Apenas 27% dos alunos participantes conseguiram responder e acertar todas as questões, durante todo o período. A não obrigatoriedade em responder as questões mostrou-se um problema na assiduidade.

O professor que ministrava a matéria nos informava quanto aos conteúdos ministrados na semana e, com essa informação conseguíamos definir as questões que seriam aplicadas. Desta forma não tivemos casos de usar questões com conteúdos ainda não abordados em sala de aula, evitando que o erro nas respostas ocorressem devido ao fato dos alunos não terem tido contato com o conteúdo. Este teste piloto foi desenvolvido com o intuito de testar a corretude das questões.

6 CONSIDERAÇÕES FINAIS

O Objetivo geral desse trabalho foi a construção de recurso educacional para desenvolver a habilidade da percepção de padrões de equívocos com aprendizes de programação. Foram desenvolvidas 63 questões abordando padrões de equívocos em C e 46 em Python, totalizando 109 questões de múltipla escolha. Desenvolvemos as questões com base nos padrões de equívocos catalogados por Bosse et al. [6]. Além das questões, construímos também um *site* onde as questões são disponibilizadas em diversos formatos e, através de um teste piloto, aplicamos questionários semanais com cinco questões com o intuito de testar a corretude das questões.

Correlacionando os padrões de equívocos com os conteúdos que eram abordados em sala de aula, podemos tornar o aprendizado mais eficiente, pois o aluno não só irá aprender como programar em C ou Python, mas também a identificar os erros comuns daquela linguagem. Aprendendo com os erros é uma técnica divulgada na literatura [13, 17, 35], levando o aluno a evitar os equívocos no futuro e, caso os cometa, saberá identificar e corrigir mais rapidamente.

Os testes com a turma piloto mostraram indícios positivos que as questões podem ser utilizadas no processo de ensino-aprendizagem, porém, sem uma obrigatoriedade em respondê-las, fez com que muitos alunos desistissem da participação pois se tornou uma tarefa a mais para eles realizarem.

6.1 Trabalhos Futuros

Como trabalhos futuros sugerimos:

- Desenvolvimento de novas questões que podem ser focadas em aspectos específicos que professor queira explorar, como por exemplo, focar somente em padrões de equívoco semânticos, ao invés de sintáticos.
- A aplicação dos questionários em uma turma real de programação, avaliando o real impacto da aplicação das questões no desenvolvimento da habilidade de percepção de padrões de equívocos.

REFERÊNCIAS

[1] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. 1977. A pattern language.
[2] Brett A Becker. 2019. A survey of introductory programming courses in Ireland. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 58–64.
[3] Jens Bennedsen and Michael E Caspersen. 2019. Failure rates in introductory programming: 12 years later. *ACM Inroads* 10, 2, 30–36.
[4] Yorah Bosse. 2020. *Padrões de dificuldades relacionadas com o aprendizado de programação.* Ph.D. Dissertation. Universidade de São Paulo.

[5] Yorah Bosse and Marco Aurelio Gerosa. 2015. Reprovações e trancamentos nas disciplinas de introdução à programação da Universidade de São Paulo: Um estudo preliminar. *XXIII WEI - Workshop sobre Educação em Computação - Em: Anais do XXXV Congresso da Sociedade Brasileira de Computação*, 1–10.

[6] Yorah Bosse, Igor Scaliante Wiese, Marco Aurélio Graciotto Silva, Nelson Lago, Leônidas de Oliveira Brandão, David Redmiles, Fabio Kon, and Marco A Gerosa. 2021. Catalogs of C and Python Antipatterns by CS1 Students. *arXiv preprint arXiv:2104.12542*.

[7] William H Brown, Raphael C Malveau, Hays W McCormick, and Thomas J Mowbray. 1998. *AntiPatterns: refactoring software, architectures, and projects in crisis*. John Wiley & Sons, Inc.

[8] Bernardo Buchweitz. 1996. Elaboração de questões de múltipla escolha. *Estudos em Avaliação Educacional* 14, 83–104.

[9] Ricardo Caceffo, Breno de França, Guilherme Gama, Raysa Benatti, Tales Aparecida, Tania Caldas, and Rodolfo Azevedo. 2017. An Antipattern documentation about misconceptions related to an introductory programming course in C. In *Technical Report 17-15*. Institute of Computing, University of Campinas, 42.

[10] Stephen Cooper, Wanda Dann, Randy Pausch, and Randy Pausch. 2003. Teaching objects-first in introductory Computer Science. *ACM SIGCSE Bulletin* 35, 1, 191–195.

[11] Edsger W Dijkstra et al. 1989. On the cruelty of really teaching Computing Science. *Commun. ACM* 32, 12, 1398–1404.

[12] Rachel D'souza, Mahima Bhayana, Marzieh Ahmadzadeh, and Brian Harrington. 2019. A Mixed-Methods Study of Novice Programmer Interaction with Python Error Messages. In *Proceedings of the Western Canadian Conference on Computing Education*. ACM, 15.

[13] Per-Erik Ellström. 2001. Integrating learning and work: Problems and prospects. *Human resource development quarterly* 12, 4, 421–435.

[14] Gerhard Fischer and Elisa Giaccardi. 2006. Meta-design: A framework for the future of end-user development. In *End user development*. Springer, 427–457.

[15] Sandy Garner, Patricia Haden, and Anthony Robins. 2005. My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*. Australian Computer Society, Inc., 173–180.

[16] Anabela Gomes and Antonio Mendes. 2014. A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, 1–8.

[17] Christian Harteis, Johannes Bauer, and Hans Gruber. 2008. The culture of learning from mistakes: How employees handle mistakes in everyday work. *International Journal of Educational Research* 47, 4, 223–231.

[18] Juha Helminen and Lauri Malmi. 2010. Jype-a program visualization and programming exercise tool for Python. In *Proceedings of the 5th international symposium on Software visualization*. ACM, 153–162.

[19] Maria Hristova, Ananya Misra, Megan Rutter, and Rebecca Mercuri. 2003. Identifying and correcting Java programming errors for introductory computer science students. *ACM SIGCSE Bulletin* 35, 1, 153–156.

[20] Tony Jenkins. 2002. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, Vol. 4. Citeseer, 53–58.

[21] AM Kristi. 2003. Problems in learning and teaching programming-a literature study for developing visualizations in the Codewitz-Minerva Project. *Codewitz Need Analysis, Institute of Software System, Tampere University of Technology, Finland*, 1–12.

[22] Cínthia Costa Kulpa, Eluza Toledo Pinheiro, and Régio Pierre da Silva. 2011. A influência das cores na usabilidade de interfaces através do design centrado no comportamento cultural do usuário. *perspectivas em Gestão & Conhecimento* 1, 1, 119–136.

[23] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin* 37, 3, 14–18.

[24] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2013. Learning Computer Science concepts with Scratch. *Computer Science Education* 23, 3, 239–264.

[25] CS De Menezes and Isaura Alcina Martins Nobre. 2002. Um ambiente cooperativo para apoio a cursos de introdução a programação. In *Congresso da Sociedade Brasileira de Computação*, Vol. 22.

[26] Mahmoud M Mhashi and ALIM Alakeel. 2013. Difficulties facing students in learning computer programming skills at Tabuk University. In *Proceedings of the 12th International Conference on Education and Educational Technology (EDU'13), Iwate, Japan*. 15–24.

[27] Jerónimo Nunes. 2015. A importância da programação informática: benefícios da aprendizagem e mais-valias competitivas. *Correio dos Açores*, 15–15.

[28] Alexandra Okada, Lynn Alves, and Daniela Barros. 2009. Moodle-estratégias pedagógicas e estudos de caso.

[29] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. In *ACM SIGCSE Bulletin*, Vol. 39. ACM, 204–223.

[30] Martinha Piteira and Carlos Costa. 2013. Learning computer programming: Study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*. ACM, 75–80.

[31] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2, 137–172.

[32] Pranay Kumar Sevella. 2013. *Determining the barriers faced by novice programmers*. Texas A&M University-Kingsville.

[33] Observatório Softex. 2009. Software e serviços de TI: a indústria brasileira em perspectiva. *Campinas:[sn]*.

[34] Renan Souza, Ricardo Caceffo, Pablo Frank-Bolton, and Rodolfo Azevedo. 2018. *An antipattern documentation about possible misconceptions related to introductory programming courses (CS1) in JAVA*. Technical Report. Technical Report IC-18–20. Institute of Computing, University of Campinas ....

[35] JN Streumer and M Kho. 2006. The world of work-related learning. In *Work-related learning*. Springer, 3–49.

[36] Phit-Huan Tan, Choo-Yee Ting, and Siew-Woei Ling. 2009. Learning difficulties in programming courses: undergraduates' perspective and perception. In *2009 International Conference on Computer Technology and Development*, Vol. 1. IEEE, 42–46.

[37] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 39–44.